Big Data and Computing Visions

www.bidacv.com

Big. Data. Comp. Vis. Vol. 3, No. 4 (2023) 160–185.

Paper Type: Original Article

A Sparrow Search Algorithm Based Hybrid Meta-Heuristic Algorithm for Population Growth Rate Prediction

Milad Shahvaroughi Farahani[1,*], Hamed Farrokhi Asl[2], Ghazal Ghasemi[3]

[1] Department of Finance, Khatam University, Tehran, Iran; m.shahvaroughi@khatam.ac.ir.
[2] Sheldon B. Lubar Business School, University of Wisconsin Milwaukee, Wisconsin, USA; farrokh5@uwm.edu.
[3] Department of Law, Islamic Azad University, Tehran, Iran; qasemi.lawyer@khatam.ac.ir.

Citation:

Shahvaroughi Farahani, M., Farrokhi Asl, H., & Ghasemi, Gh. (2023). A sparrow search algorithm based hybrid meta-heuristic algorithm for population growth rate prediction. *Big data and computing visions, 3*(4), 160-185.

Received: 06/08/2023 Reviewed: 08/09/2023 Revised: 09/10/2023 Accept: 20/11/2023

Abstract

In any economy, it is essential to monitor the rate of population change closely. Governments employ various strategies and programs to regulate population growth since different population growth rates have distinct economic consequences. This paper reveals a global trend of reduced desire to have children, with variations across countries. The paper aims to predict the population growth rate in England by employing Artificial Neural Networks (ANN) in combination with various meta-heuristic algorithms, including the Sparrow Search Algorithm (SSA). The selection of SSA and other algorithms is based on factors such as accuracy and computational efficiency. A set of 18 economic indicators serves as input variables, and a Genetic Algorithm (GA) is used for feature selection. The data used for analysis spans the most recent ten years and is presented on a monthly basis. The results indicate that SSA exhibits the lowest prediction errors for the population growth rate among the applied algorithms in this paper. The primary contribution of this study lies in the application of hybrid algorithms that combine SSA-ANN with other algorithms, such as LA. The paper also emphasizes the inclusion of influential and impactful indices as input variables to enhance prediction accuracy.

Keywords: Artificial neural network, Meta-heuristic algorithms, Sparrow search algorithm, Mayfly algorithm, Lichtenberg algorithm, Population growth rate.

1 | Introduction

Nowadays, governments employ a range of strategies and programs to manage population growth. Varied population growth rates yield distinct economic consequences. Elevated population growth rates can lead to increased production, heightened employment opportunities, higher incomes, and overall positive economic growth. Conversely, declining population growth rates can result in reduced production, decreased job opportunities, lower incomes, and negative economic growth. Extensive research and articles have explored the intricate relationship between economic and population growth [1]. It is evident that humans have always endeavored to enhance their standard of living across various dimensions, encompassing health and well-being, reduced inequality, environmental sustainability, poverty eradication, and hunger alleviation, now collectively termed Sustainable Development Goals (SDGs) [2]. It is an undeniable fact that resources are finite, whereas

Corresponding Author: m.shahvaroughi@khatam.ac.ir
https://doi.org/10.22105/bdcv.2024.426714.1170

human needs are limitless. Therefore, a growing population necessitates increased resource utilization.

In 2017, the United Nations projected a decline in global population growth rate, from +1.0% in 2020 to +0.5% in 2050 and +0.1% in 2100. As of 2020, the world's population was growing at a rate of approximately 1.05% annually (down from 1.08% in 2019, 1.10% in 2018, and 1.12% in 2017), with an estimated average population increase of 81 million people per year. Evidently, the average population growth rate has decreased globally, and the United Kingdom is no exception. The UK has experienced its lowest annual growth rate since 2003, with a population of 67.1 million in mid-2019, compared to 66.8 million people. Several factors contribute to this reduction, including changing social attitudes and the empowerment of women, increased access to contraception and modern medicine, a 23% rise in the number of people aged 65 to 84 between 2008 and 2018, and shifts in birth and mortality rates. Many countries are striving to attain the SDGs, which can enhance living standards by reducing inequality, providing access to economic growth, and more.

Table 1. Main determinants of population growth.

No	Main Determinants of Population Growth	Explanations
1	Demographic characteristics	– Initial population – Previous growth – Migration – Age and population distribution – Fertility or death rates
2	Socio-economic conditions	– Education – Educational attainments – Library /books per capita – B. economic conditions – Employment and unemployment rate – Initial economic condition – Level of wages – Housing availability – Poverty rate – crime rate – Cultural amenities
3	Natural amenities	– Climate conditions – Forest land coverage – Parks – Coastal proximity
4	Transportation accessibility	– Accessibility – Highways, railways, airport – Public transportation system – Distance variable – Rational/ province/ state dummies
5	Land use and development	– Available land – Topographical characteristics – Tax exempted lands – Built-up lands – Wetland

Predicting the population growth rate serves several pivotal purposes and has profound implications for various sectors and decision-making processes. Here are some of the key reasons why population growth rate prediction holds significant importance:

I. Urban planning: population growth prediction aids urban planners and policymakers in estimating the demand for infrastructure, public services, housing, transportation, and utilities.
II. Resource allocation: forecasting population growth enables effective resource allocation, including education, healthcare, food, water, energy, and other essential resources necessary for a growing population's sustenance.
III. Economic planning: population growth projections are crucial for economic planning, offering insights into future labor force availability, market size, consumer demand, and investment opportunities.
IV. Social services and policy: accurate population growth predictions assist governments and social service organizations in planning and providing essential services such as healthcare, education, social welfare, and elderly care. It informs policies related to family planning, immigration, and social integration.
V. Environmental impact: understanding population growth patterns is vital for assessing environmental impact and sustainability challenges. This knowledge aids in managing natural resources, land use, and environmental conservation, as well as addressing urbanization and climate change effects.

A multitude of factors influences population growth rates. Researchers have consistently sought to pinpoint the pivotal elements driving changes in population growth rates. These determinants have been categorized into five primary groups, as outlined in *Table 1* [3]. To further elaborate on the content presented in *Table 1* and to provide additional clarity, we offer *Fig. 1*. This visual representation illustrates the intricate interconnectedness between poverty incidence, population growth, and environmental degradation.

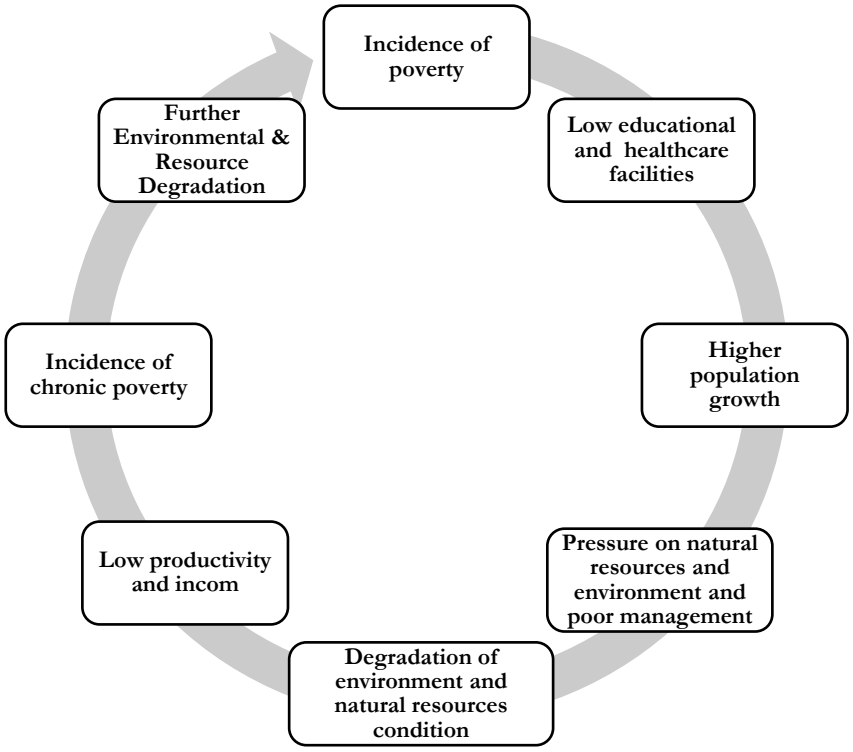

Fig. 1. Interconnection between the incidence of poverty, population growth, and degradation of the environment.

2 | Literature Review

In the realm of literature review, different approaches and categories exist. The population is commonly segmented into three categories: 1) cities, 2) counties, and 3) Metropolitan areas. Two prevalent methodologies for investigating the key determinants of population growth are the linear regression model estimated through Ordinary Least Squares (OLS) and panel data, in addition to econometric techniques.

Koomen et al. [4] presented a new modeling approach that disaggregates scenario-based national-level urban population estimates derived from the often-applied Shared Socio-economic Pathways (SSPs) to the high-resolution spatial grid (30 arcsec) representations of an urban area and local population. They wanted to describe local changes in urban areas and populations across the globe. The 2UP model presented here enhances the development of global-scale, spatially explicit population scenarios and facilitates the ongoing assessment of diverging socio-economic and environmental pathways, especially in relation to climate change. Furthermore, unlike existing, similar scale modeling approaches that either project urban land use or population distribution, this model incorporates both and simulates local urbanization dynamically. Thus, it can account for the differential growth of urban areas and populations, allowing a more refined assessment of location-specific impacts of population dynamics. Johnson et al. [5] evaluated potential drivers of population changes observed in 50 species of some of the world's most charismatic and functionally important fauna-large mammalian carnivores. Their results revealed that human socio-economic development is more associated with carnivore population declines than habitat loss or climate change. Rapid increases in socio-economic development are linked to sharp population declines, but, importantly, once development slows, carnivore populations have the potential to recover. Morgenstern et al. [6], in a review paper, aimed to predict population health using machine learning. Their analysis encompassed papers published between 1980 and 2018. The findings revealed that machine learning applications in population health predominantly concentrated on regions and diseases well-represented in traditional data sources, with limited use of big data. The authors suggested that greater utilization of big data and adherence to reporting guidelines for predictive modeling could enhance machine learning applications in population health. Munir and Shahid [7] explored the impact of demographic factors, including fertility rates and life expectancy, on the economic growth of South Asian countries. They analyzed annual panel data from four South Asian countries (Bangladesh, India, Pakistan, and Sri Lanka) spanning from 1980 to 2018. The results demonstrated a positive relationship between the real stock of capital, fertility rate, and life expectancy. In contrast, the young dependency ratio had a negative impact on economic growth in South Asian countries. *Table 2* summarizes some relevant papers in this field of study.

Table 2. Literature review about determinant factors in changing population growth rate.

Ref.	Level	Period	Region / Country	Relevant Variables	Focus	Method
[8]	Country	2008-2018	5 Countries	Growth rate, mortality, fertility, migration, growth rate of grandmothers	The components of change in population growth rates.	Derivative of a function with respect to time-based on the Horiuchi [9] model.
[10]	Urban	1951-2019	India	the temperature data	Population changes and sustainability of energy drive cooling demand-related risks in urbanized India.	five climate models under three Representative Concentration Pathways (RCPs).
[11]	Urban	1990-2020	Pakistan	Using open-source data (landsat images and landscan data)	Monitoring the population change and urban growth of four major Pakistan cities through spatial analysis of open-source data.	GIS and remote-sensing techniques were comprehensively used for change detection and directional aspects of urban growth for four decades.
[12]	Urban	2008-2018	Urban Settings	– Housing – Sanitation – Nutrition – Physical activity – Water quality – Education – Income	This study aims to systematically review key environmental determinants and respective dimensions and indicators relevant to evaluating population health in urban settings and to understand their potential implications for policies.	Preferred Reporting Items for Systematic Reviews and Meta-Analysis (PRISMA)

In the context of predicting population changes, the Sparrow Search Algorithm (SSA) can be applied by considering the population as the "food source" and the individual sparrows as the "predictors." Each predictor evaluates the suitability of a particular location or environment for the population and communicates this information with others in the group.

Through iterative iterations and interactions, the sparrows collectively converge on the most suitable locations for the population, effectively predicting where it is likely to change or grow. This process is similar to how real sparrows locate and flock to areas with abundant food resources.

The novelty of the SSA lies in its simplicity and ability to capture emergent behavior from simple rules. It does not require extensive data preprocessing or complex mathematical models, making it more accessible and computationally efficient. Additionally, its inspiration from nature provides a unique perspective on problem-solving and optimization.

However, it is essential to note that the SSA is still a relatively new concept, and its application in predicting population changes may have limitations. Its effectiveness depends on various factors, such as the quality and availability of data, the suitability of the problem domain, and the appropriateness of the algorithm parameters.

The novelty of the Lichtenberg Algorithm (LA) lies in its ability to accurately predict population changes by taking into account various factors such as birth rates, death rates, migration patterns, and socio-economic factors. Unlike traditional population prediction models that rely solely on historical data and trends, the LA incorporates real-time data and adjusts its predictions accordingly.

One key feature of the LA is its ability to account for changes in population dynamics over time. It can adapt to changing socio-economic conditions, technological advancements, and other factors that may influence population growth or decline. This makes it particularly useful for long-term population projections, as it can provide more accurate and reliable predictions compared to other models.

Another novelty of the LA is its ability to incorporate uncertainty and variability in its predictions. It uses probabilistic methods to account for the inherent randomness and unpredictability of population changes. This means that the algorithm can provide a range of possible outcomes rather than a single deterministic prediction. This is especially important when dealing with complex systems like human populations, where numerous factors can influence population changes.

Overall, the novelty of the LA lies in its ability to accurately predict population changes by incorporating real-time data, adapting to changing conditions, and accounting for uncertainty and variability. It represents an advancement in population prediction models and has the potential to provide valuable insights for policymakers, urban planners, and researchers in various fields.

3 | Methodology

To begin, the data should undergo normalization and preprocessing before entering the neural network. Multiple economic indicators serve as input variables, with an analysis spanning 18 monthly economic indicators from 2000 to 2016. Feature selection is carried out using a Genetic Algorithm (GA) to choose the most relevant and significant economic indicators [13]. Various optimization algorithms, including SSA [14], Mayfly Algorithm (MA) [15], LA [16], Particle Swarm Optimization algorithm (PSO) [17], Modified Particle Swarm Optimization algorithm (MPSO) [18], Time-Varying Accelerator Coefficients Particle Swarm Optimization algorithm (TVAC-PSO) [19], and Chimp Optimization Algorithm (ChOA) [20], are employed to optimize the network.

The Mean Square Error (MSE) is utilized as the primary loss function for evaluating the network's performance. Ultimately, the results are compared based on the magnitude of the error estimation. Each algorithm is associated with various references that provide additional information on parameters, equations, and formulas for those seeking more in-depth knowledge.

3.1 | Indicators

In this article, 17 economic indicators, including GDP rate, employment rate, earnings, and others, serve as input variables, while the target variable is an indicator representing the population growth rate. A GA is employed for feature selection, and the selection of these indicators is informed by a thorough literature review and their established relationships with population dynamics, particularly within the realm of macroeconomic studies. The specific economic indicators considered are listed in *Table 3*.

Table 3. Indicators.

No	Indicators	Type
1	GDP rate	Input variable
2	Employment rate	Input variable
3	Earnings (total activities)	Input variable
4	Population growth rate	Input variable
5	Total production	Input variable
6	import	Input variable
7	Consumer Price Index (CPI)	Input variable
8	House price changes (%)	Input variable
9	Inflation rate	Input variable
10	Health care expenditure	Input variable
11	Interest rate	Input variable
12	Household credit card rate	Input variable
13	Household loan rate	Input variable
14	Export	Input variable
15	Banking deposit rate	Input variable
16	Personal consumption expenditure	Input variable
17	Unemployment rate	Input variable
18	Population	Target

We could not find any paper about using this type of indicator that has a high explanatory power. Presenting and applying these unique indicators with high explanatory powers, along with using GA as feature selection, can make the paper interesting.

3.2 | Artificial Neural Networks

Artificial Neural Networks (ANNs) are powerful tools known for their ability to learn from data, particularly excelling in handling complex, nonlinear relationships. They operate efficiently through parallel processing and adapt well to new data, making them versatile for real-world applications. ANNs automatically extract essential features from raw data, which is beneficial for tasks like image and speech recognition. Moreover, ANNs serve as the building blocks for deep learning, revolutionizing fields like computer vision and natural language processing. Their hierarchical structure, adaptability, and generalization capabilities make ANNs invaluable for solving a wide range of problems and processing vast datasets.

However, it's important to note that the suitability and performance of ANNs depend on factors such as the problem domain, available data, network architecture design, training algorithms, and parameter selection. Proper data preprocessing, network tuning, and avoiding overfitting are key considerations in achieving optimal performance with ANNs.

A Multi-Layer Perceptron (MLP) is employed in a three-layer architecture comprising input, hidden, and output layers. The input layer consists of 17 nodes, while the number of nodes in the hidden layer is determined through iterative testing, ranging from 1 to 32 neurons to optimize performance. Notably, the output variable, representing the target, is the population growth rate. The relevant parameters are detailed in *Table 4* for reference. The proposed architecture of ANN is depicted in *Fig. 2* [15].

Table 4. Parameters.

Parameters	Explanations
Training	Back-Propagation (BP)
Optimization algorithm	Levenberg-Marquardt (LM)
Training rate	0.01
Iterations	1000
Activation function	Tan-sigmoid
	Pure line

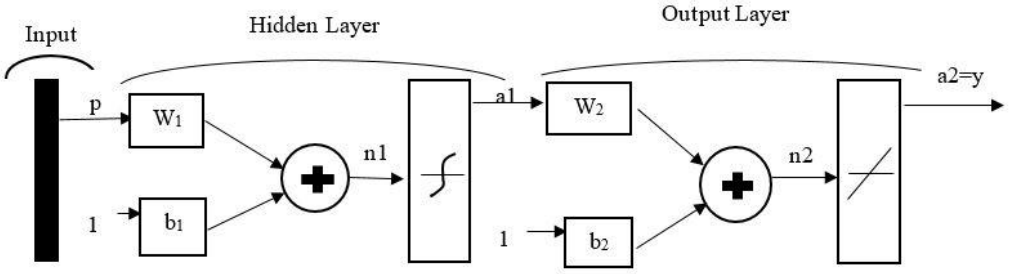

Fig. 2. Architecture of the proposed neural network [15].

70% of the data is used as training, and 30% of the data is used as validation and testing. At first, data should be normalized and prepared to enter the network using the following equation:

$$\widetilde{S_i} = \frac{(S_i - S_{min})}{S_{max} - S_{min}} . i = 1 \dots . N. \quad (1)$$

In *Eq. (1)*, numerator i is the amount of data. This formula normalizes the feature S and puts it in the range $[0.1]$. To normalize in $[-1.1]$, it is possible to use the following equation:

$$s'' = 2\frac{s_i - s_{min}}{s_{max} - s_{min}} - 1. \quad (2)$$

In general, it is possible to get a new variable s''' in $[a.b]$ by using *Eq. (3)*.

$$s''' = b - a)\frac{s_i - s_{min}}{s_{max} - s_{min}} + a. \quad (3)$$

Fine-tuning algorithm parameters is a crucial aspect of AI-based algorithms. Some parameters, like Crossover and mutation parameters, are established using literature, while others are set to their default values in the corresponding software. Key hyperparameters include the number of neurons, activation function, optimizer, learning rate, batch size, and epochs. The optimal number of neurons is determined through trial and error, with variations from one to 32 neurons tested to find the architecture that yields the best fitness value based on the loss function. For the activation function, both nonlinear and linear options are explored, incorporating functions like sigmoid, tanh, and exponential using MATLAB. The choice of activation function, learning rate, batch size, and epochs is guided by the LM optimization algorithm. Additional details on parameter tuning are provided in *Table 5*, where batch size refers to the number of samples passed through the network at once.

Table 5. Tuning the parameters and layers of ANN.

No	Parameters	Values
1	Neurons	1, 32
2	Activation	0, 5
3	Learning rate	0.01, 1
4	Batch size	100, 1000
5	Epochs	1, 1000
6	Layer 1	1, 3
7	Layer 2	1, 3

To implement and configure an ANN algorithm for predicting population changes, these steps have been followed:

I. Data collection: gather historical population data from reliable sources such as government census reports, local statistical offices, or public databases.
II. Data preprocessing: clean the data to remove any inconsistencies or missing values, and then preprocess it for use in the ANN. This may involve normalization, scaling, or encoding categorical variables.
III. Split the data: divide the dataset into training and testing sets. The training set will be used to train the ANN, while the testing set will be used to evaluate its performance.
IV. Feature selection: identify the relevant features that may influence population changes, such as demographics, economic indicators, and historical trends.
V. Building the ANN model: choose the type of ANN architecture that suits your prediction task, such as a feedforward neural network or a recurrent neural network, and decide on the number of layers, neurons, and activation functions.
VI. Training the model: use the training data to fit the ANN model to the population change prediction task. This involves adjusting the weights and biases of the network through a process called BP.
VII. Hyperparameter tuning: fine-tune the hyperparameters of the ANN, such as learning rate, number of epochs, and regularization techniques, to optimize its performance.
VIII. Evaluation: evaluate the trained ANN model using the testing dataset to assess its predictive accuracy. Common evaluation metrics include mean absolute error, MSE, and R-squared.
IX. Prediction: once the ANN model is trained and evaluated, it can be used to make predictions on new or future population data.
X. Deployment and monitoring: deploy the trained ANN model as part of a prediction system and monitor its performance over time to ensure accuracy and reliability.

Throughout this process, it's important to experiment with different configurations, architectures, and hyperparameters to achieve the best prediction results.

To implement and configure an ANN algorithm in MATLAB, you can follow these general steps. Assume you're using the neural network toolbox in MATLAB.

Algoritm 1. ANN implementation and configuration process.

Step 1. Data preparation: prepare your input data and target outputs. Make sure to normalize or standardize your input data if necessary.
Step 2. Create a neural network: use the patternnet or feedforwardnet function to create a new neural network object. For example: net = patternnet (hidden layer sizes), where hidden layer sizes is a vector specifying the number of neurons in each hidden layer.
Step 3. Configure neural network settings: set network parameters and training options using the configure function. For example: net = configure (net, inputs, targets), net. train Param. epochs = 100, net. train Param. lr = 0.01.
Step 4. Train the neural network: use the train function to train the neural network with your prepared data: net = train (net, inputs, targets).
Step 5. Test the neural network: after training, you can use the net to make predictions on new data: outputs = net (inputs).
Step 6. Evaluate the model: evaluate the performance of the trained neural network using appropriate metrics and validation techniques.

3.3 | Genetic Algorithm

GA is a global search technique that mimics the principles of biological evolution [21]. In this application, binary coding with 22 bits is employed, where 17 bits represent the presence or absence of a variable (indicated by 1 and 0, respectively). The remaining 5 bits (22 - 17 = 5) are utilized to determine the number of hidden layer neurons, offering a range of 32 possibilities (25 = 32). The initial population is generated stochastically with a size of 20 individuals. The individuals with the lowest MSE are chosen to participate in crossover and mutation operations. The algorithm runs for 100 epochs and is later extended to 1000 for improved results. Similar to the ANN, the data is divided into 70% for training and 30% for validation and testing. A training rate of 0.01 is initially set, gradually decreasing over time with repetitions. For additional details, please consult *Table 6*. The 20 best individuals will be selected as new generations, and this process

will repeat and continue until the end of the desired conditions and termination criteria. Mutation and crossover operation is in *Fig. 3* [22].

Table 6. GA parameters.

Output Error	Output Activation Function	Input Activation Function	Mutation Rate	Crossover Rate	Number of Generation	Population Size	Max Itr
MSE	Tan-Sigmoid	Simple linear	0.1	0.9	50	20	2000
Selection parents			Mutation		Crossover		
Roulette wheel method			Binary method		One-point method		

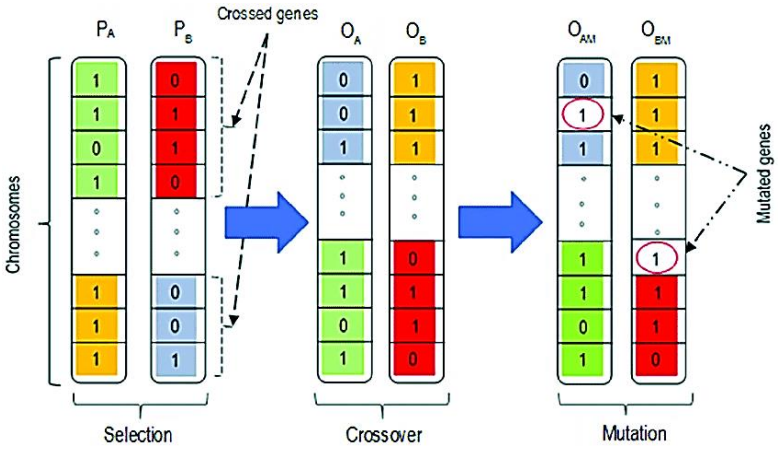

Fig. 3. Crossover and mutation operations [22].

Crossover in GAs combines two parent strings to generate offspring, essentially creating new solutions. The probability of crossover typically falls within the range of 0.8 to 0.95. Mutation is employed to introduce variability and generate entirely new solutions.

While GAs offer advantages for feature selection, their performance relies on factors like the choice of genetic operators, fitness function design, population size, and termination criteria. Model selection aims to identify a neural network topology that minimizes error on new data. Two types of model selection algorithms are used: order selection algorithms and input selection algorithms. Order selection algorithms determine the optimal number of hidden neurons, while input selection algorithms identify the optimal subset of input variables. For this application, the chosen order selection algorithm is incremental order. This method begins with the minimum order and incrementally adds a specified amount of perceptron in each iteration.

Table 7. Order selection algorithm parameters.

Parameters	Description	Value
Minimum order	Number of minimum hidden perceptrons to be evaluated.	1
Maximum order	Number of maximum hidden perceptrons to be evaluated.	10
Step	Number of hidden perceptrons added in each iteration.	1
Trials number	Number of trials for each neural network.	3
Tolerance	Tolerance for the selection error in the training of the algorithm.	0.01
Selection loss goal	Goal value for the selection error.	0
Maximum selection failures	Maximum number of iterations at which the selection error increases.	5
Maximum iterations number	Maximum number of iterations to perform the algorithm.	1000
Maximum time	Maximum time for the order selection algorithm.	3600
Plot training error history	Plot a graph with the training error of each iteration.	True
Plot selection error history	Plot a graph with the selection error of each iteration.	True

Algoritm 2. GA pseudo-code.

Step 1. Initialize the population
- *Generate an initial population of chromosomes (potential solutions) randomly or using a specific heuristic.*
- *Each chromosome represents a candidate solution, encoded as a string of genes (binary or real values).*

Step 2. Calculate the fitness of each chromosome
- *Evaluate the fitness of each chromosome based on a fitness function, which reflects how good the solutions are.*
- *Assign a fitness value to each chromosome.*

Step 3. Repeat the following steps until a termination condition is met (e.g., maximum number of iterations reached).

Step 4. Selection
- *Select parent chromosomes from the population based on their fitness. Common methods include roulette wheel selection, tournament selection, or rank-based selection.*

Step 5. Crossover
- *Perform crossover (recombination) between selected parent chromosomes to create offspring (new solutions).*
- *Apply a crossover operator (e.g., single-point crossover, two-point crossover, uniform crossover) to combine genetic material from parents and generate offspring with different traits.*

Step 6. Mutation
- *Apply a mutation operator to introduce random changes in some genes of the offspring population.*
- *This step adds diversity to the population and helps in exploring new regions of the search space.*

Step 7. Evaluate the fitness of offspring
- *Evaluate the fitness of the newly created offspring (child chromosomes) using the fitness function.*

Step 8. Replace
- *Select individuals for the next generation, which will form the population for the next iteration.*
- *Replace less fit individuals from the current population with the newly created offspring.*

Step 9. Increment the generation counter.

Step 10. Return the best solution found or the solution with the highest fitness.
- *(Optional) perform post-processing or analysis of the obtained solutions if needed.*

Additional details regarding the order selection algorithm parameters are available in the appendix. The parameters of the order selection algorithm are outlined in *Table 7*. Additionally, the pseudo-code of GA in this research has been shown in *Algoritm 2*.

Here's an example of how you might implement these steps in MATLAB:

Algoriem 3. GA implementation and configuration process.

Step 1. Example fitness function (you should replace this with your own): fitness function = @ (features) your custom fitness function (features, data, labels).

Step 2. GA options: options = gaoptimset ('population size', 20, 'generations', 50, 'stall gen limit', 15).

Step 3. Call the GA: [selected features, fval] = GA (fitness function, number of features, options).

Step 4. Use the selected features with your learning algorithm: replace your custom fitness function with your actual fitness function, and modify the options based on your problem's requirements.

3.4 | Sparrow Search Algorithm

The Sparrow algorithm, also known as the Social-Driven Optimization (SDO) algorithm, draws inspiration from the behavior of sparrows to address search and optimization problems.

There are two distinct types of sparrows within this algorithm: producers and scroungers [23]. Each type possesses unique characteristics. Producers are abundant in energy and actively seek out nutrient-rich areas for the Scroungers. In the presence of predators, sparrows emit warning chirps. If these alarms surpass a safety threshold, producers lead all Scroungers to safety. Sparrows can transition between being producers and Scroungers based on the availability of nutrient-rich areas, although the ratio of producers to Scroungers remains fixed in the population. Starving Scroungers may venture to other regions in search of more food and energy. Scroungers often tail producers to locate food and, at times, observe producers for predation opportunities. Sparrows adjust their positions within the group in response to perceived threats.

Those in precarious situations strive to reach safer areas within the group, while those already safe move randomly to maintain proximity. This dynamic can be represented in a matrix illustrating the positions of sparrows.

$$\begin{bmatrix} x_{1.1} & x_{1.2} \cdots & \cdots & x_{1.d} \\ x_{2.1} & x_{2.2} \cdots & \cdots & x_{2.d} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n.1} & x_{n.2} \cdots & \cdots & x_{n.d} \end{bmatrix}, \tag{4}$$

where

n: The number of sparrows.

d: The dimension of the variables to be optimized.

The fitness value of all sparrows can be expressed using the following vector:

$$F_x = \begin{bmatrix} f([x_{1.1} & x_{1.2} \cdots & \cdots & x_{1.d})] \\ f([x_{2.1} & x_{2.2} \cdots & \cdots & x_{2.d})] \\ \vdots & \vdots & \vdots & \vdots \\ f([x_{n.1} & x_{n.2} \cdots & \cdots & x_{n.d})] \end{bmatrix}, \tag{5}$$

where

n: The number of sparrows.

F_x: The value of each row shows the fitness value of individuals.

Each sparrow, such as the producer and scrounger, has a location. The location of the producer can be updated as follows:

$$X_{ij}^{t+1} = \begin{cases} X_{ij}^{t} \cdot \exp\left(\dfrac{-i}{\alpha \cdot iter_{max}}\right), & \text{if} R_2 < ST, \\ X_{ij}^{t} + Q \cdot L, & \text{if} R_2 \geq ST, \end{cases} \tag{6}$$
$$j = 1, 2, \ldots, d,$$

where

t: The current iteration.

X_{ij}^{t}: The value of the jth dimension of the jth sparrow at iteration t.

$iter_{max}$: A constant with the largest number of iterations.

a ($\alpha \in [0.1]$): A random number.

R_2 ($R_2 \in [0.1]$) and ST ($ST \in [0 \cdot 5.1]$): The alarm value and the safety threshold, respectively.

Q: A random number with a normal distribution.

L: A matrix of $1 \times d$ for which each element inside is 1.

$R_2 < ST$: It means there are no predators around and a wider search mode.

$R_2 \geq ST$: It means a predator has been seen, and they should fly to other safe areas.

Scroungers update their positions using the following expression:

$$X_{ij}^{t+1} = \begin{cases} Q \cdot \exp\left(\dfrac{X_{worst}^{t} - X_{ij}^{t}}{i^2}\right), & \text{if } i > n/2, \\ X_P^{t+1} + \left|X_{ij}^{t} - X_P^{t+1}\right| \cdot A^{+} \cdot L, & \text{Otherwise,} \end{cases} \tag{7}$$

where

X_P: The optimal position occupied by the producer.

X_{worst}: The current global worst location.

A: Matrix of $1 \times d$ for which each element inside is randomly assigned 1 or -1.

A^+: $A^T(AA^T)^{-1}$.

$i > n/2$: It means that the i^{th} scrounger with the worse fitness value is most likely to be starving.

The rate of sparrows that are aware of danger is 10% to 20% of the total population. Their initial position is random. This position can be modeled by using the following expression:

$$X_{ij}^{t+1} = \begin{cases} X_{best}^{t} + \beta \cdot \left|X_{ij}^{t} - X_{best}^{t}\right|, & \text{if } f_i > f_g, \\ X_{ij}^{t} + K \cdot \left(\dfrac{\left|X_{ij}^{t} - X_{worst}^{t}\right|}{(f_i - f_w) + \varepsilon)}\right), & \text{if } f_i = f_g, \end{cases} \tag{8}$$

where

X_{best}: The current global optimal location.

β: The step size control parameter ($\beta \sim N[0.1]$).

K: A random number ($K \epsilon$ [−1.1].

f_i: The fitness value of the present sparrow.

$f_g$and f_w: The current global best and worst fitness values, respectively.

ε: The smallest constant so as to avoid zero-division-error.

$f_i > f_g$: It means that the sparrow is at the edge of the group.

$f_i = f_g$: Sparrows are aware of dangers and need to move closer to the others.

K: The direction in which the sparrow moves and is also the step size control coefficient.

The model can summarize SSA steps by using the following pseudo-codes in *Algoritm 4* [16].

Algoritm 4. SSA pseudo-code [16].

Input:
G: The maximum iterations.
PD: The number of producers.
SD: The number of sparrows who perceive the danger.
R_2: The alarm values.
n: The number of sparrows.
Initialize a population of n sparrows and define its relevant parameters.
Output: X_{best}, f_g.
1: While (t < G).
2: Rank the fitness values and find the current best individual and the current worst individual.
3: R_2 = Rand (1).
4: for i = 1: PD.
5: Using *Eq. (3)* update the sparrow's location.
6: End for.
7: For i = (PD + 1): n.
8: Using *Eq. (4)* update the sparrow's location.
9: End for.
10: For l = 1: SD.
11: Using *Eq. (5)* update the sparrow's location.
12: End for.
13: Get the current new location.
14: If the new location is better than before, update it.
15: t = t + 1.
16: End while.

17: Return: X_{best}, f_g.

SSA implementation and configuration for prediction of population changes can be approached in the following way:

Algoritm 5. SSA implementation and configuration process.

Step 1. Define the problem: clearly define the problem of predicting population changes. This may involve specifying the variables that contribute to population changes, such as birth rates, death rates, migration, etc.

Step 2. Formulate the objective function: create an objective function that represents the population change prediction problem. This function should take the relevant variables as input and output a measure of how well a particular set of input variables predicts population changes.

Step 3. Implement the SSA: write or obtain an implementation of the SSA in a programming language of your choice. The algorithm mimics the behavior of sparrows foraging for food and can be used to optimize the parameters of the objective function.

Step 4. Integrate objective function with SSA: integrate the objective function into the SSA so that the algorithm can optimize the input variables to minimize or maximize the objective function, depending on the problem.

Step 5. Configuration and parameter tuning: Configure the SSA by setting parameters such as the population size, maximum number of iterations, convergence criteria, and other relevant parameters. Tweak these parameters based on the problem-specific requirements and the behavior of the algorithm.

Step 6. Data collection and preprocessing: gather historical population data and other relevant data, preprocess and clean the data as needed for input into the prediction model.

Step 7. Evaluation and validation: use historical data to train and validate the prediction model. Evaluate the performance of the model by comparing its predictions against actual population changes.

Step 8. Refinement and iteration: based on the evaluation results, refine the model and the configuration of the SSA. Iterate on the process to improve the accuracy of the population change predictions.

Remember that while the SSA can be a powerful optimization tool, it should be used in conjunction with domain-specific knowledge and other statistical or machine-learning techniques for prediction tasks. Also, the availability of appropriately structured data greatly influences the success of predictive modeling. In this paper, we have tried to predict population changes using two almost new algorithms,

SSA and LA. So, for other algorithms, you can refer to their references. The LA implementation and configuration are available in the appendix (*Fig. 18*).

3.5 | Evaluation of the Performance of the Models

After obtaining optimal solutions, the accuracy of the model is checked. So, 3 criteria such as accuracy, precious, and sensitivity, are used for comparability [24]. These criteria are calculated using the following equations:

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}. \tag{9}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}. \tag{10}$$

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{Total}}. \tag{11}$$

Where

- *TP: True Positive.*
- *FN: False Negative.*
- *TN: True Negative.*
- *TP: True positive.*

In machine learning, there is a matrix which is called a "Confusion Matrix." This matrix shows the performance of an algorithm, that is, the difference between predicted and actual values.

4 | Findings and Results

4.1 | Data Statistics

Firstly, as we mentioned earlier, data need to be normalized between -1 and 1 and made ready as input variables. Monthly data and 18 economic indicators are used to predict population changes from Feb-2000 to Dec-2016. Different data about indices are obtained through three main websites:

I. The World Bank (https://data.worldbank.org/indicator/sp.pop.totl).
II. OECD (https://data.oecd.org/pop/population.htm).
III. FRED economic data (https://fred.stlouisfed.org/categories).

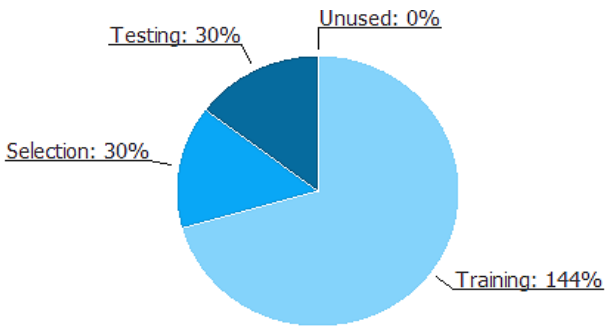

Fig. 4. Instances pie chart.

There are two kinds of variables: 1) input variables, and 2) target. As it is clear, the number of instances is 204. *Fig. 4* details the uses of all the instances in the data set. The total number of instances is 204. The number of training instances is 144 (70.6%), the number of selection instances is 30 (14.7%), the number of testing instances is 30 (14.7%), and the number of unused instances is 0 (0%).

Table 8 shows the value of the correlations between all input and target variables. The maximum correlation (0.995613) is the yield between the input variable CPI and the target variable population.

Table 8. Population correlation table.

Variables	Type	Population
CPI	Linear	0.995613
Personal consumption expenditure	Linear	0.986354
Earning (total activities)	Linear	0.967218
Interest rate	Linear	-0.921116
Total production	Linear	-0.836179
Import	Linear	0.791401
Household loan rate	Linear	-0.769222
Bank deposit rate	Linear	-0.750759
Population monthly growth rate	Linear	0.739825
Export	Linear	0.666692
Health care expenditure	Linear	0.565657
Unemployment rate	Linear	0.449978
Housing price change (%)	Linear	-0.444087
Household credit card rate	Linear	0.309567
Employment rate	Linear	0.230633
Inflation rate	Linear	0.019849
GDP normalized	Linear	-0.01755

4.2 | Artificial Neural Networks

In this part, the network has been trained without using any algorithms or GA. Three steps are used to complete the part: 1) finding the best architecture (designing), 2) training the network, and 3) validation and testing.

Table 9. Best network architecture.

ID	Architecture	# of Weights	Fitness	Train Error	Validation Error	Test Error	AIC	Correlation	R-Squared	Stop Reason
1	[17-3-1]	58	0.014875	51.218632	70.50119	67.226173	-24.775463	0.999546	0.99904	All iterations done
2	[17-43-1]	818	0.026988	22.51347	46.323139	37.053329	1380.145995	0.999881	0.999752	All iterations done
3	[17-27-1]	514	0.015612	51.199024	61.312561	64.054077	887.170922	0.999395	0.998727	All iterations done
4	[17-17-1]	324	0.018325	27.910839	53.400375	54.570042	422.232177	0.999828	0.99964	All iterations done
5	[17-11-1]	210	0.018116	37.767017	57.393887	55.200855	236.571118	0.999718	0.99942	All iterations done
6	[17-23-1]	438	0.016004	27.820793	46.225163	62.485294	649.779774	0.999836	0.999661	All iterations done
7	[17-20-1]	381	0.016473	34.194599	54.911961	60.705578	564.659544	0.999751	0.999479	All iterations done
8	[17-14-1]	267	0.021231	33.215752	49.521759	47.100254	332.593453	0.999765	0.999512	All iterations done
9	[17-15-1]	286	0.024444	23.101105	41.661892	40.910267	319.75332	0.999877	0.999743	All iterations done
10	[17-16-1]	305	0.01991	29.967207	44.45879	50.224957	394.184574	0.999818	0.999617	All iterations done

70% and 30% of data are used for training, validation, and testing, respectively. *Table 9* shows the best architecture of the network after 1000 iterations. Blue shows the best architecture, which means 43 neurons are hidden in layers with a maximum R square of 0.999752. This architecture has been achieved after 10 trials and errors. The best network error during iterations has been depicted in *Fig. 5*. The network properties are summarized in *Table 10*.

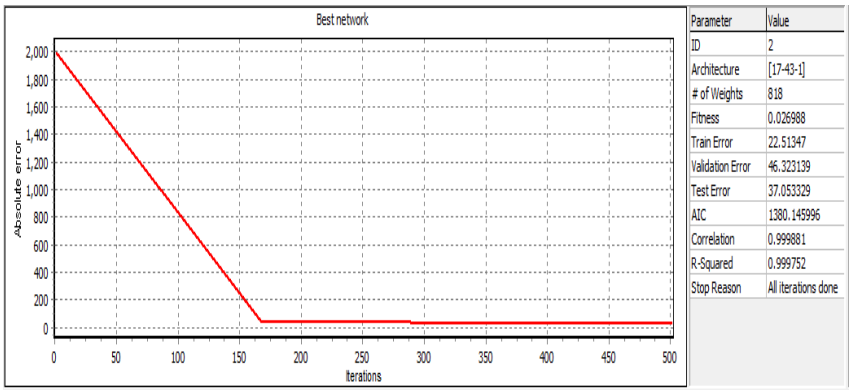

Fig. 5. Best network error.

Table 10. Network properties.

Parameter	Value
Input activation FX	Logistic
Output name	Daily death cases (CLOSE)
Output error FX	Sum-of squares
Output activation FX	Logistic

After finding the best architecture, the network needs training. *Figs. 6* and *7* show the training and validation errors and their improvement during iterations, respectively. LM is used as a training algorithm. The network has done 342 iterations because there has been no error improvement since then. A sharp decline can be seen in network error after 340 iterations. Training parameters are also illustrated in *Table 11*.

Fig. 6. Dataset error.

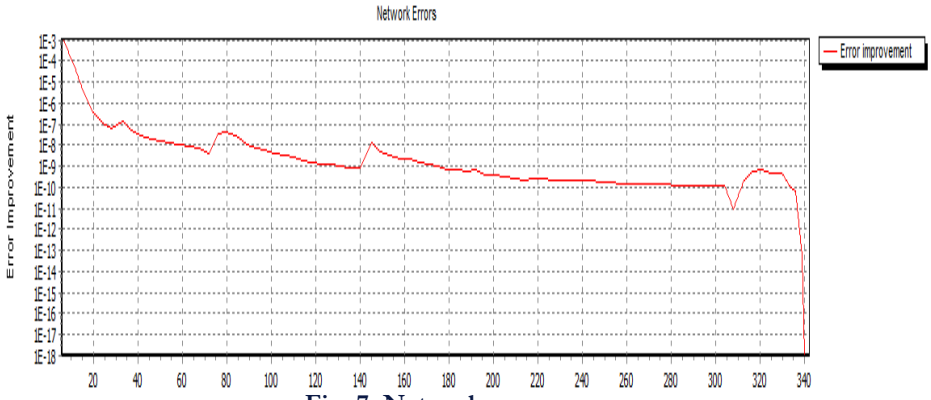

Fig. 7. Network errors.

Table 11. Training parameters.

	Training	Validation
Absolute error	0.85616	184.881953
Network error	2.16E-08	0
Error improvement	1.39E-21	
Iteration	342	
Training speed, ite/sec	21.111125	
Architecture	[17-8-1]	
Training algorithm	LM	
Training stop reason	No error improvement	

Now, it is the time to train the network. So, 30% of the data is used to train the network. *Fig. 8* presents the testing error plot. More details and summary statistics have been presented in *Table 12*.

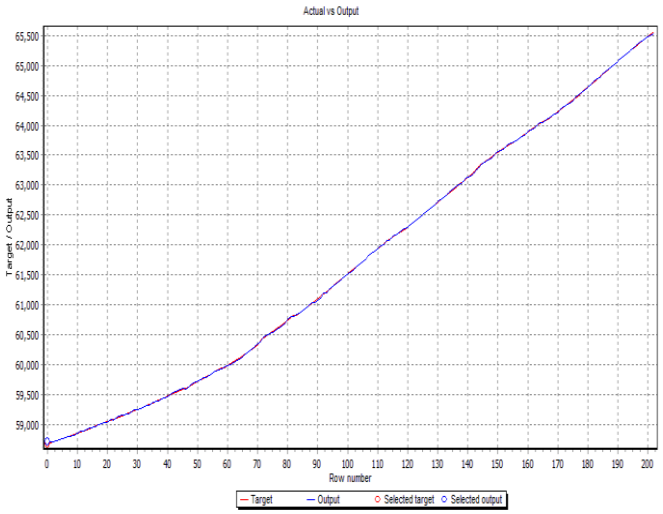

Fig. 8. Testing error (actual vs. output).

Table 12. Summary statistics.

	Target	Output	AE	ARE
Mean	61680.94391	61680.948762	10.349246	0.000168
Std. Dev	2108.179553	2107.542537	6.085667	0.0001
Min	58684	58716.483463	0.072803	0.000001
Max	65563.762161	65530.779094	32.983067	0.000554
Correlation	0.999984			
R-Squared	0.999968			

4.3 | Genetic Algorithm

Fig. 9 shows the error history for the different subsets during the incremental order selection process. The blue line represents the training error, and the orange line symbolizes the selection error. *Table 13* also shows the order selection results by the incremental order algorithm. They include some final states from the neural network, the error function, and the order selection algorithm.

Table 13. Order selection results.

Parameters	Value
Optimal order	1
Optimum training error	0.0165844
Optimum selection error	0.014907
Iterations number	10
Elapsed time	0:01

Fig. 9. Incremental order error plot.

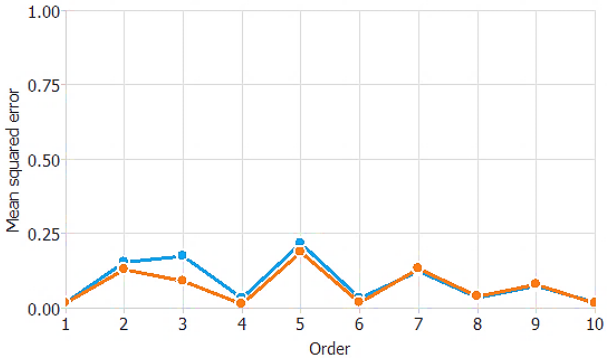

Fig. 9. Incremental order error plot.

In this study, the GA is utilized for feature selection to address issues stemming from redundant inputs in some datasets, which can impact the neural network's error. Input selection aims to identify the optimal subset of inputs for achieving the best model error. Details regarding the parameters of the input selection algorithm can be found in the appendix.

Fig. 10 illustrates the error history throughout the GA input selection process. The blue line represents the training error, commencing at an initial value of 0.0365473 and concluding at 0.0342988 after 100 generations. The orange line denotes the selection error, with an initial value of 0.0257535 and a final value of 0.0161364 after 100 generations.

Table 14 presents the outcomes of the input selection achieved by the GA, encompassing the final states of the neural network, the error functions, and details about the input selection algorithm.

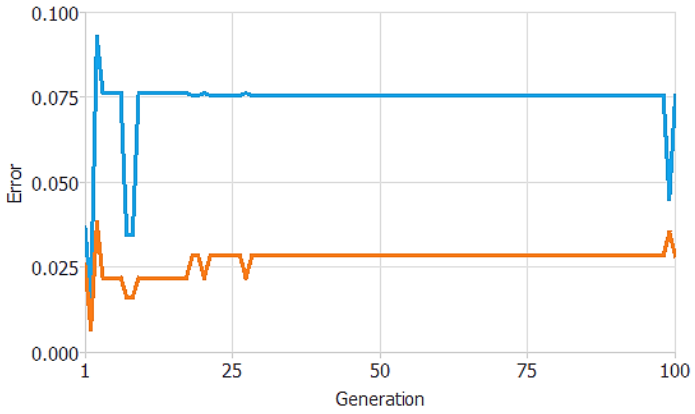

Fig. 10. GA error plot.

Table 14. GA results.

Parameters	Value
Optimal number of inputs	11
Optimum training error	0.0342988
Optimum selection error	0.0161364
Generations number	100
Elapsed time	00:00

A visual representation of the resulting deep architecture is provided below, comprising a sealing layer, a neural network, and an un-scaling layer. The diagram employs yellow circles to denote scaling neurons, blue circles for perceptron neurons, and red circles for un-scaling neurons. With 11 inputs and 1 output, the complexity is characterized by the presence of a single hidden neuron. Notably, it is evident that all variables are selected, suggesting their collective importance.

Figs. B1 and *B2* in the appendix depict linear regression for the scaled output population, showcasing predicted values versus actual ones represented as circles. The grey line signifies the best linear fit. Lastly, *Table 15* compiles all data errors for various use cases.

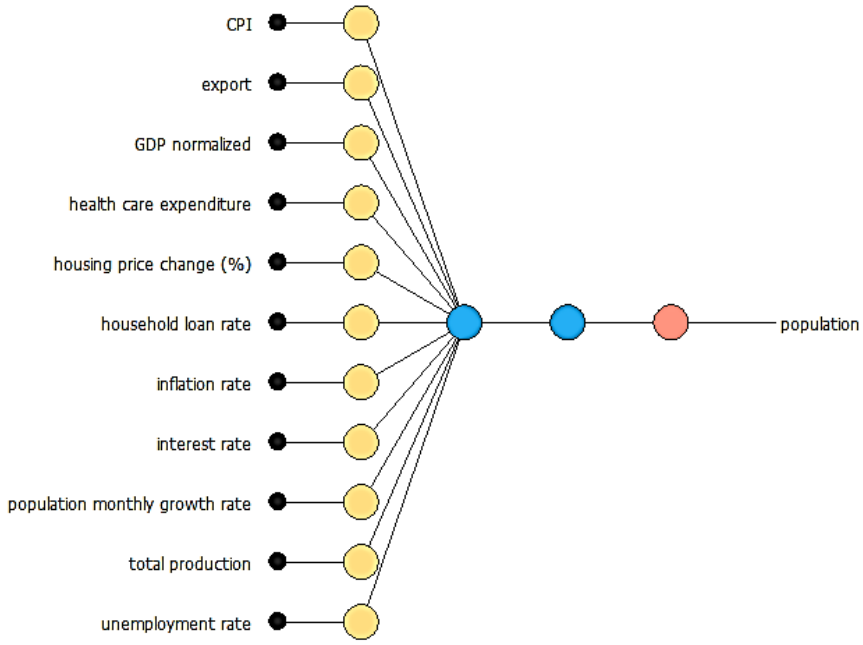

Fig. 11. Final architecture.

Table 15. GA error table.

Criteria	Training	Selection	Testing
Sum squared error	1.17868	0. 413017	0.420068
MSE	0.00950548	0.0103254	0.0105017
Root MSE	0.0974961	0.101614	0.102478
Normalized squared error	0.0987531	0.12939	0.128097
Minkowski error	3.67556	1.27223	0.29128

4.4 | Mayfly Algorithm

In this section, MA is employed to train the network and enhance error reduction. The relevant parameters for this process are detailed in *Table 16*. Two objective functions, namely "sphere" and "rastrigin," are utilized.

Table 16. MA parameters.

No	Parameter	Quantity
1	Problem size	[150]
2	Upper bound	10
3	Lower bound	-10
4	Max-Iteration	2000
5	Population size (male and female)	$Npop_m = 20$ $Npop_f = 20$
6	Inertia weight (g)	0.8
7	Inertia weight damping ratio (gdamp)	1
8	Personal learning coefficient (a1)	1
9	Global learning coefficient (a2, a3)	a2 = 1.5 a3 = 1.5

The error plot, reflecting the progress achieved after 2000 iterations, is displayed in *Figs. 12* and *13*. After 2000 iterations, the error rates for "sphere" and "rastrigin" are 1.6845e-18 and 12.936, respectively.

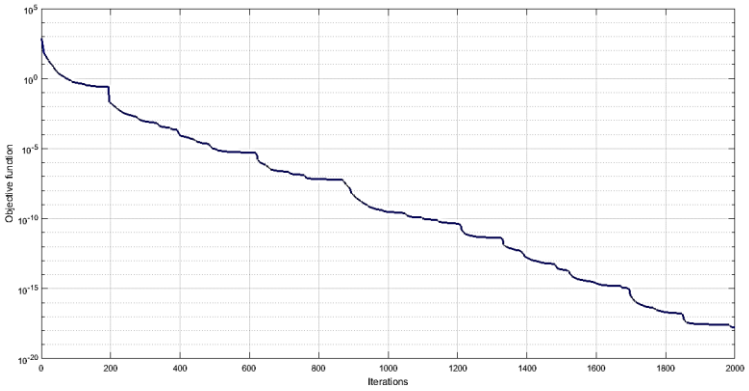

Fig. 12. Error plot (sphere function).

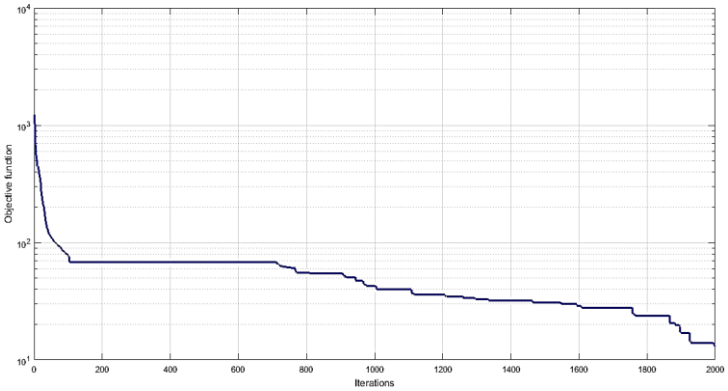

Fig. 13. Error plot (rastrigin).

4.5 | Lichtenberg Algorithm

Table 17 provides the parameters associated with the LA. The corresponding error plot and the Lichtenberg figure can be found in *Figs. 14* and *15*, respectively.

Table 17. LA parameters.

No	Parameters	Quantity
Bounds		
1	Upper bounds	[5 5]
2	Lower bounds	[-5 5]
3	Problem dimension	d= length (LB)
LA Parameters		
4	Population size	pop=20
5	Max number of iterations	100
6	Number of particles	100.000
7	Stick probability	1
8	Creation radius	150
9	Refinement (use after running the algorithm once	0.2
10	Creating new lichtenberg figures in each iteration	M=0
11	Initial point (only for the first population)	x0= (LB+UB)/2

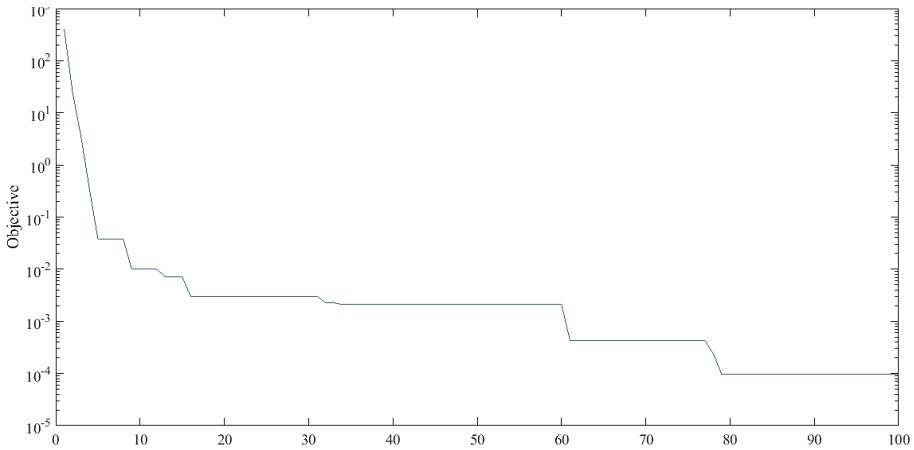

Fig. 14. Error plot.

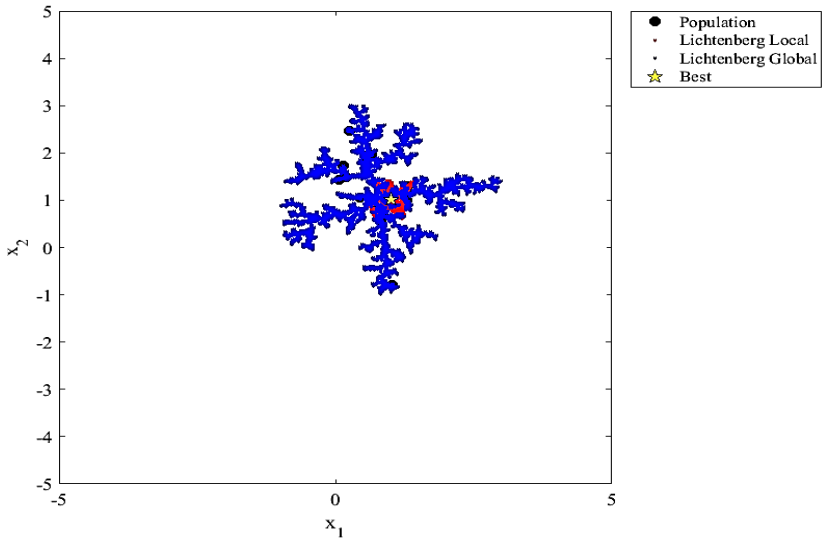

Fig. 15. Lichtenberg figure.

Ultimately, the outcomes for Var1 and Var2 are 1.0096 and 1.0195, respectively. The best fitness achieved is 9.5532e-05, and the process took 56.451015 seconds to complete.

4.6 | Sparrow Search Algorithm

The following algorithm is SSA. So, the considered parameters are summarized in *Table 18*. The 30 best solutions obtained by SSA are obtained, the average is 3.8892e-174, and the best optimal value of the objective function found by SSA is 0. Finally, the statues of other considered algorithms are shown in *Fig. 17*.

Table 18. SSA parameters.

No	Parameters	Quantity
Bounds		
1	Upper bounds	100
2	Lower bounds	-100
3	Max-iterations	1000
4	Search agent number	100
5	Fmin	0
6	Dim	30

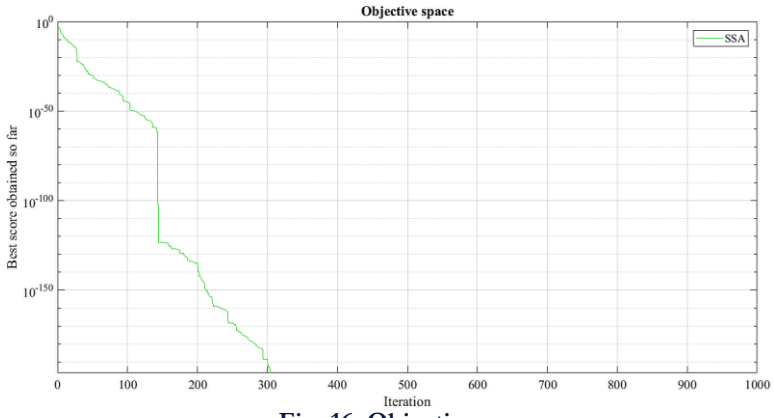

Fig. 16. Objective space.

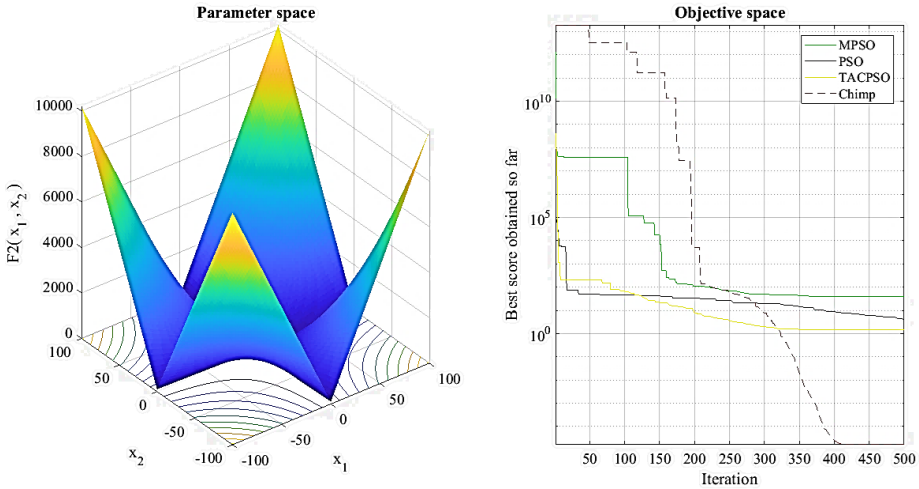

Fig. 17. Parameters and objective space.

The best optimal values of the objective function found by TACPSO, PSO, MPSO, and ChOA are 1.4797, 4.0839, 40.1701, and 1.6474e-05, respectively. It should be noted that these algorithms have the same fitness functions, MSE.

4.7 | Comparing the Results

In this part, the results are compared based on error. So, *Table 19* shows the least and the most error for each algorithm. It is essential to highlight that the same fitness function is applied across all the algorithms, ensuring a fair basis for comparison.

Indeed, the results indicate that SSA and MPSO exhibit the lowest and highest errors among the considered algorithms. However, it's crucial to recognize that there isn't a single "best" algorithm, as each algorithm operates on distinct mechanisms and calculations, making them suitable for different scenarios and problem types. The choice of algorithm should align with the specific problem and its characteristics.

Table 19. Arrange algorithms based on MSE.

No	Algorithm	MSE
1	SSA	3.8892e-174
2	MA	1.6845e-18
3	ANN	2.16e-08
4	ChOA	1.6474e-05
5	LA	9.5532e-05
6	GA	0.00950548
7	TACPSO	1.4797
8	PSO	4.0839
9	MPSO	40.1701

After different calculations, the models should be compared based on their accuracy and different criteria. Among these methods, the SSA algorithm has the highest true positive, and this can increase the predictability. The following table presents the considered criteria, which means predictive performance metrics. As it is clear, between these algorithms, SSA has better performance since it has higher predictability in all three indices.

Table 20. Arrange algorithms based on predictive performance metrics.

	Recall (%)	Precision (%)	Accuracy (%)
SSA	95.23	92.30	88.23
LA	93.58	91.14	85.78
MA	91.89	89.94	83.33
ANN	90.16	88.70	80.88
ChOA	88.39	87.43	78.43
GA	86.59	86.11	75.80
TACPSO	84.74	84.74	73.52
PSO	82.85	83.33	71.07
MPSO	80.92	81.71	68.62

5 | Conclusion and Remarks

Population dynamics have far-reaching implications for a country, affecting its economic, social, political, and cultural landscapes. Population growth can lead to an increase in the labor force, higher production, more significant savings, enhanced consumption, and overall welfare. Governments formulate policies based on diverse societal needs, addressing issues such as healthcare for the elderly and employment opportunities for university graduates. Economic indicators play a pivotal role in guiding these policies. This article focuses on predicting population changes in the United Kingdom from January 2000 to December 2016. It begins with the utilization of ANN as a predictive model and subsequently employs a GA for feature selection. Custom indicators are identified, and various metaheuristic algorithms, including SSA and MPSO, are utilized. Results demonstrate that SSA and MPSO yield the least and most error, respectively. The benefits of employing novel metaheuristic algorithms include accelerated calculations, reduced model complexity, enhanced network accuracy, and user-friendly model application. Future research recommendations involve exploring additional novel metaheuristic algorithms like the Bald Eagle or Golden Eagle, along with bio-inspired optimization algorithms such as the improved bat algorithm. It is also suggested that a comparative analysis between statistical models and AI-based models be conducted. Controlling and decreasing the population growth rate can be achieved through various strategies:

I. Education and awareness: providing education and raising awareness about family planning, reproductive health, and the advantages of smaller family sizes can empower individuals to make informed decisions. Access to comprehensive sex education and reproductive healthcare is crucial.
II. Improved healthcare and child survival: enhancing healthcare infrastructure, particularly in developing regions, can reduce population growth. Better healthcare, immunization, nutrition, and clean water access lower child mortality rates, influencing family planning decisions.
III. Access to family planning services: ensuring the availability and affordability of contraceptives allows individuals to plan their families effectively. Accessible family planning services promote voluntary family planning and reduce unintended pregnancies.
IV. Empowering women: gender equality, women's empowerment, education, economic opportunities, and decision-making access can decrease population growth. Empowered women tend to have smaller families and make choices aligned with their well-being.
V. Economic development: fostering economic growth and improving living standards can indirectly reduce population growth. Economic progress offers better education, healthcare, and job opportunities, influencing family size choices.
VI. Policy interventions: government policies are critical for managing population growth. Policies supporting family planning, offering incentives for smaller families, and addressing socio-economic

factors can be effective. Subsidies, taxation policies, small family incentives, and even controversial measures like mandatory birth control programs are considered.

The effectiveness of population control strategies depends on regional cultural, social, and economic contexts. A comprehensive approach, combining multiple strategies, is often necessary for successful population management.

Prediction of population changes can be helpful for policymakers in several ways:

I. Planning for infrastructure and services: policymakers can use population predictions to plan for the future needs of a growing population. This includes building new schools, hospitals, transportation networks, and other essential infrastructure. By anticipating population changes, policymakers can ensure that adequate resources are allocated to meet the demands of the population.
II. Resource allocation: population predictions can help policymakers allocate resources efficiently. For example, they can determine where to invest in healthcare, education, housing, and other public services based on projected population growth or decline. This enables them to prioritize and distribute resources effectively to areas with the greatest need.
III. Economic development: population predictions can guide policymakers in shaping economic development strategies. They can identify potential areas of growth and investment based on population trends. Suppose a particular region is predicted to experience significant population growth. In that case, policymakers can focus on attracting businesses and industries to that area, which can create jobs and stimulate economic growth.
IV. Social welfare programs: policymakers can use population predictions to design and implement social welfare programs. By understanding the demographic changes in their population, they can adapt social security, healthcare, and other social programs to meet the evolving needs of the population. For instance, if the population is aging, policymakers can develop policies and programs that address the specific healthcare and retirement needs of older adults.
V. Environmental planning: population predictions can also inform policymakers about the potential environmental impacts of population changes. By understanding how population growth or decline might affect natural resources, energy consumption, waste management, and other environmental factors, policymakers can develop sustainable and environmentally friendly policies.

Overall, population predictions provide valuable insights to policymakers, enabling them to make informed decisions, allocate resources effectively, and plan for the future needs of their population.

Acknowledgments

I want to convey my heartfelt gratitude to Dr. Razavi Haji-Agha for his tremendous support and assistance in the completion of my paper.

Funding

Any organization does not fund this study

Conflicts of Interest

All co-authors have seen and agreed with the contents of the manuscript, and there is no financial interest to report. We certify that the submission is original work and is not under review at any other publication.

References

[1] Rudel, T. (2023). Population, development and tropical deforestation: a cross-national study. In *The causes of tropical deforestation* (pp. 96–105). Routledge.

[2] Eisenmenger, N., Pichler, M., Krenmayr, N., Noll, D., Plank, B., Schalmann, E., … & Gingrich, S. (2020). The sustainable development goals prioritize economic growth over sustainable resource use: a critical reflection on the SDGs from a socio-ecological perspective. *Sustainability science*, *15*, 1101–1110.

[3] Calka, B., Orych, A., Bielecka, E., & Mozuriunaite, S. (2022). The ratio of the land consumption rate to the population growth rate: A framework for the achievement of the spatiotemporal pattern in Poland and Lithuania. *Remote sensing*, *14*(5), 1074. https://doi.org/10.3390/rs14051074

[4] Koomen, E., Van Bemmel, M. S., Van Huijstee, J., Andrée, B. P. J., Ferdinand, P. A., & Van Rijn, F. J. A. (2023). An integrated global model of local urban development and population change. *Computers, environment and urban systems*, *100*, 101935. https://doi.org/10.1016/j.compenvurbsys.2022.101935

[5] Johnson, T. F., Isaac, N. J. B., Paviolo, A., & González-Suárez, M. (2023). Socioeconomic factors predict population changes of large carnivores better than climate change or habitat loss. *Nature communications*, *14*(1), 74. https://www.nature.com/articles/s41467-022-35665-9

[6] Morgenstern, J. D., Buajitti, E., O'Neill, M., Piggott, T., Goel, V., Fridman, D., … & Rosella, L. C. (2020). Predicting population health with machine learning: a scoping review. *BMJ open*, *10*(10), e037860.

[7] Munir, K., & Shahid, F. S. U. (2021). Role of demographic factors in economic growth of South Asian countries. *Journal of economic studies*, *48*(3), 557–570.

[8] Canudas-Romo, V., Shen, T., & Payne, C. F. (2022). The components of change in population growth rates. *Demography*, *59*(2), 417–431.

[9] Horiuchi, S. (1991). Assessing the effects of mortality reduction on population ageing. *Population bulletin of the united nations*, *31*(32), 38–51.

[10] Yaduvanshi, A., Singh, R., & Kumar, R. (2022). Population changes and sustainability of energy drive cooling demand related risks in urbanized India. *Energy and buildings*, *260*, 111891. https://doi.org/10.1016/j.enbuild.2022.111891

[11] Aslam, R. W., Shu, H., & Yaseen, A. (2023). Monitoring the population change and urban growth of four major Pakistan cities through spatial analysis of open source data. *Annals of gis*, *29*(3), 355–367. https://doi.org/10.1080/19475683.2023.2166989

[12] Salgado, M., Madureira, J., Mendes, A. S., Torres, A., Teixeira, J. P., & Oliveira, M. D. (2020). Environmental determinants of population health in urban settings. A systematic review. *BMC public health*, *20*, 1–11.

[13] Shahvaroughi Farahani, M., & Razavi Hajiagha, S. H. (2021). Forecasting stock price using integrated artificial neural network and metaheuristic algorithms compared to time series models. *Soft computing*, *25*(13), 8483–8513.

[14] Xue, J., & Shen, B. (2020). A novel swarm intelligence optimization approach: sparrow search algorithm. *Systems science & control engineering*, *8*(1), 22–34.

[15] Zervoudakis, K., & Tsafarakis, S. (2020). A mayfly optimization algorithm. *Computers & industrial engineering*, *145*, 106559. https://doi.org/10.1016/j.cie.2020.106559

[16] Pereira, J. L. J., Francisco, M. B., Diniz, C. A., Oliver, G. A., Cunha Jr, S. S., & Gomes, G. F. (2021). Lichtenberg algorithm: A novel hybrid physics-based meta-heuristic for global optimization. *Expert systems with applications*, *170*, 114522. https://doi.org/10.1016/j.eswa.2020.114522

[17] Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: a systematic review. *Archives of computational methods in engineering*, *29*(5), 2531–2561.

[18] Ang, K. M., Chow, C. E., El-Kenawy, E.-S. M., Abdelhamid, A. A., Ibrahim, A., Karim, F. K., … & Lim, W. H. (2022). A modified particle swarm optimization algorithm for optimizing artificial neural network in classification tasks. *Processes*, *10*(12), 2579. https://doi.org/10.3390/pr10122579

[19] Khandelwal, M. K., & Sharma, N. (2023). A survey on particle swarm optimization algorithm. *International conference on communication and computational technologies* (pp. 591–602). Springer.

[20] Khishe, M., & Mosavi, M. R. (2020). Chimp optimization algorithm. *Expert systems with applications*, *149*, 113338. https://doi.org/10.1016/j.eswa.2020.113338

[21] Sohail, A. (2023). Genetic algorithms in the fields of artificial intelligence and data sciences. *Annals of data science*, *10*(4), 1007–1018.

[22] Singh, G., & Gupta, N. (2022). A study of crossover operators in genetic algorithms. In *Frontiers in nature-inspired industrial optimization* (pp. 17–32). Springer. https://link.springer.com/chapter/10.1007/978-981-16-3128-3_2

[23] Zhang, H., & Zhang, Y. (2023). An improved sparrow search algorithm for optimizing support vector machines. *IEEE access*, *11*, 8199–8206.

[24] Madhu, M. S., & others. (2022). Detection of liver disorder using quadratic support vector machine in comparison with rbf svm to measure the accuracy, precision, sensitivity and specificity. *2022 international conference on innovative computing, intelligent communication and smart electrical systems (ICSES)* (pp. 1–7). IEEE.

Appendix A

The Lichtenberg Algorithm, also known as the lightning fractal or Lichtenberg figure algorithm, is a process used to simulate the branching patterns seen in electrical discharge, such as lightning.

```MATLAB
% Define the dimensions of the grid and initialize
it
width = 400;
height = 400;
grid = zeros(height, width);

% Set the starting point
x = round(width/2);
y = height;

% Number of iterations
iterations = 10000;

% Perform the simulation
for i = 1:iterations
    % Mark the current position as visited
    grid(y, x) = 1;

    % Move to a random adjacent cell
    options = [x-1, y; x+1, y; x, y-1; x, y+1];
    valid = options(:,1) > 0 & options(:,1) <= width
& ...
            options(:,2) > 0 & options(:,2) <=
height;
    options = options(valid,:);
    if isempty(options)
        % If there are no valid options, restart
from the top
        x = round(width/2);
        y = height;
    else
        % Move to a random adjacent cell
        idx = randi(size(options,1), 1);
        x = options(idx, 1);
        y = options(idx, 2);
    end
end

% Display the resulting figure
imagesc(grid);
colormap(gray);
axis equal;
axis off;
```

Fig. A1. A simple version of the Lichtenberg Algorithm in MATLAB.